The Next Decade of the CPAchecker Development Process

Dirk Beyer and **Philipp Wendler**
Contributions by
Thomas Lemberger and Marian Lingsch-Rosenfeld

SoSy-Lab @ LMU Munich

CPA'24 2024-09-09

CPAchecker Development History

- 2007: CPAchecker project started
- 2008: SVN repository created
- 2010: Issue tracker Trac set up
- 2014: Official git mirror created
- 2018: Issue tracking moved to GitLab.com,
  started to use GitLab merge requests for code reviews

CPAchecker Development History

- 2007: CPAchecker project started
- 2008: SVN repository created
- 2010: Issue tracker Trac set up
- 2014: Official git mirror created
- 2018: Issue tracking moved to GitLab.com,
  started to use GitLab merge requests for code reviews

So it is time for something new?

SVN+Git ⇒ Git

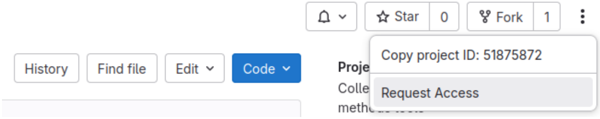
- Give up SVN repository, move to pure Git setup
- Road blocks now removed:
  VerifierCloud/BenchCloud and BuildBot
  Thanks BenchCloud team and Marian!
- ETA: this **October**
- Seamless switch expected:
  just update git remote config and start using `git push`

Future Project Resources

- Main repository: **GitLab.com**
- `trunk` renamed to `main`
- GitHub remains read-only mirror
- `https://svn.sosy-lab.org/software/cpachecker.git` gone
- Links to `https://svn.sosy-lab.org/software/cpachecker` redirected to GitLab
  Tells us if broken link found!

Requirement from You

- Create account on gitlab.com and
- "Request Access" on
  `https://gitlab.com/sosy-lab/software/cpachecker/`



- Sign up for `cpachecker-devel` mailing list
  cf. `https://cpachecker.sosy-lab.org/contact.php`

Development Policies and Changes

Continue: **Work in Official Repo – not Forks**

- For everyone (regular contributors, students, etc.)
- Advantages for all sides:
  - Visibility (projects and code)
  - Permanent availability
  - **CI resources!**
- No enforcement
  (but we will keep an eye on how it goes)

Change: **Pushing to Main Branch** (I)

For non-regular contributors (e.g., students):

- No pushes to main branch
- Have to create branch + MR
- Require 1 approval on MR from regular contributor (typically their mentor who should already now review their code)
- Require green CI
- May merge themselves afterwards

Change: **Pushing to Main Branch** (II)

For regular contributors:

- Still allowed (for now)...
- ... but **branches + MRs encouraged**!
- MR approval not (yet) required...
- ... but **code reviews encouraged**!
- To be decided in the future whether rules are made stricter.

Change: **Pushing to Main Branch** (II)

For regular contributors:

- Still allowed (for now)...
- ... but **branches + MRs encouraged**!
- MR approval not (yet) required...
- ... but **code reviews encouraged**!
- To be decided in the future whether rules are made stricter.

Requires cooperation from you:

- Use of MRs
- Actively seek reviewers for your MRs
  (you typically know best who would be a good fit)
- Provide reviews for others
  (even if not perfect
  – some review is better than no review!)

Change: **Pushing to Main Branch** (Reasons)

- Code quality
- Knowledge sharing (in both directions!)
- Green CI on main branch
- Creating branch + MR is now less effort
  (`git push origin HEAD:new-branch` and click on link;
  GitLab can auto-merge MRs once pipelines finishes)
- Most embarrassing errors so far checked
  with pre-commit hook (~~CPAChecker~~), but no longer

Continue: **Work Close to Main Branch**

- Value early merges!
- Less effort overall:
  - Less merge conflicts
  - Global refactorings automatically applied
  - New CI checks immediately active
- BuildBot tests only main branch!
- Months old branches typically an anti-pattern

Continue: **No Force Pushes**

- No negative experiences with SVN in this regard
- Ensures permanent code availability

$\Rightarrow$ Force pushes remain forbidden.

Branch-Management Policy

Already hundreds of branches and more to come
– what to do with old stuff?

SVN branches (currently existing ones):

- Were guaranteed to remain available after deletion
- Developers might have linked to them
- Plan: keep them available

Git branches (created in the future):

- Not visible which branch points to what at specific date (even without force pushes)
- Should not be linked to anyway
- Plan:
  - Delete merged branches
  - Keep unmerged branches (even if abandoned/obsolete)

Handling Obsolete Branches

- Plan: do nothing special
- Alternatives exist but not convincing
- Large list of branches no real problem
- Suggestions welcome
  (How do other large projects handle this?)

Thank you for all your contributions and cooperation!